REMARKS

This is a full and timely response to the final Official Action sent **March 24, 2009** (the "Office Action" or "Action"). Reconsideration of the application in light of the above amendments and the following remarks is respectfully requested.

Claim Status:

Claims 2-20 have previously been cancelled without prejudice or disclaimer and claims 21-39 have been previously added. By the foregoing amendment, claim 1 has been amended. No new matter has been added. Thus, claims 1 and 21-39 are currently pending for further action.

Claim Objections:

In the recent Office Action, claim 1 was objected to because of a minor informality. While Applicant does not necessarily agree that claim 1 was previously objectionable, claim 1 has been amended as requested by the Examiner to expedite the prosecution of this application. This amendment does not, and is not intended to, change or narrow the scope of claim 1. Following entry of this amendment, the objection to claim 1 may be reconsidered and withdrawn.

Prior Art

Rejections under 35 U.S.C. §103(a):

Claims 1 and 21-39 were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,988,261 to Sokolov et al. (hereinafter "Sokolov") in view of U.S.

Patent No. 6,014,519 to Egashira (hereinafter "Egashira"). For at least the following reasons, this rejection should be reconsidered and withdrawn.

Claim 1:

Independent claim 1 now recites:

> A method of optimizing the performance of an interpreter based runtime system, the runtime system including a virtual machine, the virtual machine adapted to run an application in the context of the runtime environment, the method comprising:
> augmenting a bytecode set of the virtual machine with semantically enriched opcodes, thereby constituting an application domain-specific virtual machine;
> optimizing the virtual machine based on semantics of the application to be run on the virtual machine, with at least a portion of the semantically enriched opcodes being specific to the application;
> performing a quantitative trade-off between execution time and memory space to determine effective semantically enriched opcodes and encoding the semantically enriched opcodes into interpreter action codes based upon the trade-off;
> analyzing frequently executed bytecodes and encoding the semantically enriched opcodes into interpreter action codes of the instruction set of the virtual machine to efficiently decode the frequently executed bytecodes;
> optimizing a translation by the interpreter action codes of the semantically enriched opcodes into a native code according to a system state, said system state being represented by at least one symbolic variable; and
> statically embedding the semantically enriched opcode to optimize execution of the interpreter-based runtime system.

To begin, Sokolov and Egashira fail to teach or suggest "augmenting a bytecode set of the virtual machine with semantically enriched opcodes, thereby constituting an application domain-specific virtual machine." (Claim 1). The Office Action asserts that Sokolov teaches the claimed "semantically enriched opcodes." (Action, p. 7) (citing to Sokolov, Figs. 2B, 12A, 13C, and col. 6, line 55 to col. 6, line 26). Applicant's specification clearly teaches that semantically enriched opcodes incorporate information relating to one or more of hardware characteristics, an application, and/or the relationship between the hardware and the application. In contrast, Sokolov teaches that "macro instructions" are automatically generated simply by operating on the byte code stream. (Sokolov, Figs. 2A, 2B, 3, 4; col. 5,

line 55 to col. 7, line 19). For example, Sokolov teaches that no semantic context needs to be considered, but operates by simply identifying a sequence of two or more Java instructions within a Java instruction stream. (Sokolov, col. 7, line 7-10). This sequence of two or more Java instructions is then automatically replaced with a macro instruction. (Sokolov, Figs. 3 and 4, col. 7, lines 7-19). Because the "macro instructions" taught by Sokolov do not account for the semantic context within which the Java code operates, the "macro instructions" cannot be the "semantically enriched opcodes" taught by the Applicant.

In response, the recent Office Action claims that a semantically enriched opcode as used in claim 1 must be defined as "being augmented/replaced from a sequence of bytecode" in light of the fact that Applicant's specification teaches that "[o]nline sEc-opcode embedding was adopted to replace the sequence of bytecode comprising the sEc-opcode with a single corresponding sEc-opcode." (Action, pp. 2-3) (citing to Applicant's specification, p. 14, lines 3-4). Applicant disagrees, noting that the Examiner here has asserted that a specific embodiment taught in Applicant's specification is a definition of a sEc-opcode (semantically enriched opcode) while utterly failing to acknowledge Applicant's express definition of the term sEc-opcode elsewhere in the specification.

Under Applicant's express definition, a "sEc-opcode is a flow sensitive maximal computational sequence of Java bytecode *deduced, for speed, from the Java application's static or dynamic behaviour*." (Applicant's specification, p. 9, lines 3-5) (emphasis added). Thus, the semantically enriched opcode recited in claim 1 must be deduced from the context of the static or dynamic behavior of a Java application. This definition must be respected, as the Office is not at liberty to substitute its own definition of a term for one expressly stated in Applicant's specification. *See Markman v. Westview Instruments*, 116 S. Ct. 1384 (1996) ("The meaning of words used in the claims is determined by the meaning given to those

words in the specification."); *Lear Siegler, Inc. v. Aeroquip Corp.*, 733 F.2d 881, 888-89, 221 U.S.P.Q. 1025 (Fed. Cir. 1984) ("The inventor may his own lexicographer").

Because the "macro instructions" taught by Sokolov are created by simply identifying a sequence of two or more Java instructions within a Java instruction stream and not from the static or dynamic behavior of the Java application itself, the Action has failed to demonstrate that Sokolov teaches or suggests the "semantically enriched opcodes" recited in claim 1.

Further, nowhere do Sokolov or Egashira teach or suggest an "application domain specific virtual machine" as recited in claim 1. Applicant's specification specifically defines an "application domain specific virtual machine" as created by an "adaptive code set for a particular Java Virtual Machine" which is application-specific or application tuned. (Applicant's specification, p. 7, lines 28-32). This results in an application domain specific virtual machine which is driven from the bottom-up from the application's static or dynamic behavior as opposed to a top-down/fixed pattern repository approach. (*Id.*).

The recent Action asserts that a Java virtual machine customized by the aforementioned "macro instructions" taught by Sokolov reads on this subject matter. (Action, p. 3) (citing to Sokolov, col. 4, lines 3-15; col. 6, lines 23-26; and col. 7, lines 51-59). In contrast, Sokolov teaches top-down/fix pattern repository (*see* Sokolov, Appendix A for an example of the fix pattern repository) which dictates the replacement of specific sequences of bytecode instructions irrespective of the application or semantic context. (Sokolov, Figs. 3 and 4, col. 7, lines 7-19). Because Sokolov fails to teach or suggest a virtual machine specific to an application domain, Sokolov *cannot* teach the "application domain specific virtual machine" recited in claim 1. Consequently Sokolov does not teach or suggest "augmenting a bytecode set of the virtual machine with semantically enriched opcodes, thereby constituting an application domain-specific virtual machine." (Claim 1).

Sokolov and Egashira also fail to teach or suggest "optimizing the virtual machine based on semantics of the application to be run on the virtual machine, with at least a portion of the semantically enriched opcodes being specific to the application." (Claim 1). With reference to this subject matter, the recent Action cites to essentially the same portions of Sokolov as cited above with respect to "augmenting a bytecode set of the virtual machine with semantically enriched opcodes." (Action, p. 7) (citing to Sokolov, Figs. 2B, 2C, and 5; col. 4, lines 3-15; col. 6, lines 23-26; col. 7, lines 51-59; col. 8, line 62 to col. 9, line 17). As amply demonstrated above, nowhere does Sokolov teach or suggest the "semantically enriched opcodes" recited in claim 1, or that at least a portion of the semantically enriched opcodes are "specific to the application." (Claim 1). Accordingly, Sokolov *cannot* teach or suggest "optimizing the virtual machine based on semantics of the application to be run on the virtual machine, with at least a portion of the semantically enriched opcodes being specific to the application." (*Id.*).

Additionally, Sokolov does not teach or suggest "performing a quantitative comparison between execution time and memory space to determine effective semantically enriched opcodes." The recent Office Action asserts that because the macro instructions taught by Sokolov "executes [sic] faster (fewer instructions) and occupy less memory space (fewer instructions)" that Sokolov teaches this subject matter. (Action, p. 7) (citing to Sokolov, col. 2, line 64 to col. 3, line 9; col. 5, line 55 to col. 6, line 9). Applicant respectfully disagrees. The fact that Sokolov teaches a virtual machine that executes faster and occupies less memory space is no indication that a proactive "quantitative trade-off" has been calculated between execution time and memory space with the specific goal of "determin[ing] effective semantically enriched opcodes" or that the semantically enriched opcodes have been encoded "into interpreter action codes *based on the trade-off*." (Claim 1)

(emphasis added). Moreover, because Sokolov fails to teach or suggest the semantically enriched opcodes of claim 1, Sokolov *cannot* teach or suggest this subject matter. Hence, Sokolov has not been shown to teach or suggest "performing a quantitative trade-off between execution time and memory space to determine effective semantically enriched opcodes and encoding the semantically enriched opcodes into interpreter action codes based upon the trade-off." (Claim 1).

The Office Action also asserts that Sokolov teaches "optimizing the translation by the interpreter action codes of the semantically enriched opcodes according to a system state." In addition to not teaching "semantically enriched opcodes", Sokolov does not teach or suggest "interpreter action codes" or "optimization translation…according to a system state." (Claim 1). The Office Action has interpreted the translation process of this step as synonymous with the encoding process of the previous step. (Action, p. 2). Applicant notes that this conclusion is incorrect, as Applicant's specification teaches that "interpreter action codes" map the bytecode sequence in the semantically enriched opcode into optimized portable native C code for the target machine. (Applicant's specification, p. 10, lines 25-31). Accordingly, the Office Action has not demonstrated that Sokolov or Egashira teaches or suggests "optimizing a translation by the interpreter action codes of the semantically enriched opcodes into a native code according to a system state, said system state being represented by at least one symbolic variable." (Claim 1).

The Office Action concedes that Sokolov does not disclose "performing a quantitative trade-off between execution time and memory space to determine effective semantically enriched opcodes and encoding the semantically enriched opcodes into interpreter action codes based upon the trade-off." (Action, p. 8). Consequently, the Office Action cites to Egashira. However, Egashira fails to remedy the shortcomings of Sokolov.

Specifically, neither Sokolov nor Egashira teach or disclose "semantically enriched opcode" or "encoding semantically enriched opcodes into interpreter action codes based upon the trade-off."

The Supreme Court recently addressed the issue of obviousness in *KSR Int'l Co. v. Teleflex Inc.*, 127 S.Ct. 1727 (2007). The Court stated that the *Graham v. John Deere Co. of Kansas City*, 383, U.S. 1 (1966), factors still control an obviousness inquiry. Under the analysis required by *Graham v. John Deere*, 383 U.S. 1 (1966) to support a rejection under § 103, the scope and content of the prior art must first be determined, followed by an assessment of the differences between the prior art and the claim at issue in view of the ordinary skill in the art. In the present case, the scope and content of the prior art, as evidenced by Sokolov and Egashira, did not include the claimed subject matter, particularly "a semantic enriched opcode," "application domain specific virtual machine," "optimizing the virtual machine based on the semantics of the application to be run on the virtual machine," "performing a quantitative comparison between execution time and memory space to determine effective semantically enriched opcodes" or "optimizing the translation by the interpreter action codes of the semantically enriched opcodes according to a system state."

The differences between the cited prior art and the claimed subject matter are significant because the claimed method provides substantial improvements in the performance of virtual machines by leveraging semantic information. (Applicant's specification, p. 14, lines 3-20). Thus, the claimed subject matter provides features and advantages not known or available in the cited prior art. Consequently, the cited prior art will not support a rejection of claim 1 and its dependent claims under 35 U.S.C. § 103 and *Graham*.

Claim 30:

Independent claim 30 now recites:

A system for optimizing performance of an interpreter based runtime system, the runtime system including a virtual machine, the system comprising:
application code;
an embedded processor;
a virtual machine configured to translate said application code into native machine code compatible with said embedded processor;
a detection module, said detection module being configured to analyze said application code to identify code segments that could be efficiently represented as *semantically enriched opcodes*, *at least a portion of said semantically enriched opcodes being specific to said application*;
an embedding module, said embedding module being configured to *embed said semantically enriched opcodes in said application*;
a code generation module, said code generation module being configured to generate optimized action code for translating said semantically enriched opcodes *according to symbolic states, each of said symbolic states being represented by at least one symbolic variable*; and
a build module configured create an *application domain-specific virtual machine* by incorporating said optimized action code and a bytecode set comprising said semantically enriched opcodes into said virtual machine.
(Emphasis added)

As discussed above, Sokolov and Egashira do not teach or suggest, either separately or in combination, the recited "semantically enriched opcodes" or that the "semantically enriched opcodes being specific to the application." (Claim 30). Consequently, Sokolov and Egashira *cannot* teach or suggest the following elements recited in claim 30:

- "a detection module . . . configured to analyze said application code to identify code segments that could be efficiently represented as *semantically enriched opcodes*, *at least a portion of said semantically enriched opcodes being specific to said application,*"
- "an embedding module . . . configured to *embed said semantically enriched opcodes in said application*,"

- "a code generation module . . . configured to generate optimized action code for *translating said semantically enriched opcodes according to symbolic states*," and
- "a build module configured to create an application domain-specific virtual machine by incorporating said optimized action code *and a byteset comprising said semantically enriched opcodes into said virtual machine*."

(Emphasis added).

Again, under the analysis required by *Graham v. John Deere*, 383 U.S. 1 (1966) to support a rejection under § 103, the scope and content of the prior art must first be determined, followed by an assessment of the differences between the prior art and the claim at issue in view of the ordinary skill in the art. In the present case, the scope and content of the prior art, as evidenced by Sokolov and Egashira, did not include the claimed subject matter. Specifically, the scope and content of the prior art did not include any of the detection module, the embedding module, the code generation module, or the build module recited in claim 30.

The differences between the cited prior art and the claimed subject matter are significant because the claimed method provides substantial improvements in the performance of virtual machines by leveraging semantic information. (Applicant's specification, p. 14, lines 3-20). Thus, the claimed subject matter provides features and advantages not known or available in the cited prior art. Consequently, the cited prior art will not support a rejection of claim 1 and its dependent claims under 35 U.S.C. § 103 and *Graham*.

Additionally, various dependent claims of the application recite subject matter that is further patentable over the cited prior art. Specific, non-exclusive examples follow.

Claims 21 and 31

Claim 21:

> The method of claim 1, further comprising analyzing an application code using static optimization, the static optimization comprising parsing the application code to identify at least one repeated sequence of bytecodes and replace the at least one repeated sequence of bytecodes with a semantically enriched opcode.

Similarly, claim 31 recites:

> The system of claim 30, wherein said detection module is configured to analyze said application code using static optimization, said static optimization comprising parsing said application code to identify at least one repeated sequence of bytecodes and replace said at least one repeated sequence of bytecodes with a semantically enriched opcode.

Claims 21 and 31 are patentable for at least the same reasons given above for the patentability of independent claims 1 and 30. Additionally, nowhere does Sokolov teach or suggest the recited "static optimization" of application code. (Claim 21). In contrast, Sokolov teaches that "a Java macro instruction generator" reads and operates on "a stream of Java Bytecode ***during Java Bytecode verification***." (Sokolov, col. 4, lines 1-15) (emphasis added). Operations on a "stream of Java Bytecode during Java Bytecode verification" cannot be reasonably interpreted as the claimed "***static*** optimization." (Claim 21) (emphasis added). Consequently, the cited prior art will not support a rejection of claims 21 and 31 under 35 U.S.C. § 103 and *Graham*. For at least this additional reason, the rejection of claims 21 and 31 should be reconsidered and withdrawn.

Claims 23 and 33

Claim 23:

The method of claim 1, further comprising:
discovering at least one repetitive computational sequence used in a symbolic state; and
generating a semantically enriched opcode corresponding to the at least one repetitive computational sequence used in the symbolic state.

Similarly, claim 33 recites:
The system of claim 30, wherein said generation module is further configured to:
discover at least one repetitive computational sequence used in a said symbolic state; and
generate a semantically enriched opcode corresponding to the at least one repetitive computational sequence used within said symbolic state.

As discussed above, Sokolov and Egashira do not teach or suggest, either separately or in combination, the recited "semantically enriched opcodes" or "symbolic states." Further, Sokolov and Egashira do not teach or suggest the claimed relationship between the semantically enriched opcodes and symbolic state, namely, "semantically enriched opcode corresponding to the at least one repetitive computational sequence used within said symbolic state." (Claim 23).

The recent Action asserts that Solokov's "predetermined number of times" associated with a loop in a "method for generating Java macro instructions" teaches the symbolic state recited in claims 23 and 33. (Action, p. 5). Applicant strongly disagrees, noting that the process of generating Java macro instructions taught by Solokov is not a component of the application being compiled for the virtual machine. As such, a state of a method for generating Java macro instructions is utterly irrelevant to a symbolic state of a virtual machine that will be executing the code. Hence, the Action has failed to establish that Solokov or Egashira teaches or suggests the subject matter of claims 23 and 33.

Consequently, the cited prior art will not support a rejection of claims 23 and 33 and under 35 U.S.C. § 103. For at least this additional reason, the rejection of claims 23 and 33 should be reconsidered and withdrawn.

Claims 24 and 34

Claim 24:

> The method of claim 23, wherein the symbolic state comprises at least one of: control flow, data flow, entry points, and operational arguments.

Similarly, claim 34 recites:

> The system of claim 33, wherein said symbolic state comprises at least one of: control flow, data flow, entry points, and operational arguments.

The Action asserts that "Solokov explicitly teaches the symbolic state comprises at least one of: control flow, data flow, entry points, and operational arguments." (Action, p. 6) (citing to Solokov, col. 5, line 55 to col. 6, line 26). In the first place, Solokov does not "explicitly" teach the symbolic state for the simple reason that the words "symbolic state" do not appear anywhere in its entire disclosure.

The Applicant is at a loss as to the relevance of the cited sections to the claimed subject matter. Nowhere within the cited sections is a "symbolic state" taught or suggested. Further nowhere within the cited sections is a relationship between a "symbolic state" and any of a "control flow, data flow, entry points, and operational arguments" taught or suggested. Consequently, the cited prior art will not support a rejection of claims 24 and 34 under 35 U.S.C. § 103. For at least these additional reasons, the rejection of claims 24 and 34 should be reconsidered and withdrawn.

According to the Supreme Court, the Examiner is required to provide an explicit analysis as to how the cited prior art teaches or suggests all the features of a claim. "To facilitate review, this [the Examiner's] analysis should be made explicit." (*KSR International Co. v. Teleflex, Inc.*, 550 U.S. ___ (2007). As demonstrated above, the Examiner has failed to specifically demonstrate how or where Sokolov teaches or suggests the features of a "symbolic state" that comprises at least one of a "control flow, data flow, entry points, and operational arguments." (Claim 24). Therefore, under the standard of *KSR*, no *prima facie* case of obviousness has been made as to claims 24 and 34. For at least this reason, the rejection of claims 24 and 34 should be withdrawn.

Claims 25 and 35

Claim 25:

> The method of claim 1, further comprising optimizing native code output by the virtual machine using a global optimizing compiler.

Similarly, claim 35 recites:

> The system of claim 30, further comprising a global optimizer compiler configured to optimizing native code output by said virtual machine.

Claims 25 and 35 are patentable for at least the same reasons given above for the patentability of claims 1 and 30. Additionally, Sokolov and Egashira do not teach or suggest, either separately or in combination, the recited "optimization of native code output by the virtual machine using a global optimizing compiler." (Claim 25).

With respect to this subject matter, the Office Action Again cites to Solokov's teaching that "different Java macro instructions are generated" and that "a compiler can be used with different Java macro instructions in different applications." (Action, p. 6).

Applicant notes that the Java macro instructions taught by Solokov are not "native code ***output*** by said virtual machine." (Claim 25) (emphasis added). Rather, they are coded instructions that are ***executed*** by the virtual machine. This distinction cannot be overlooked.

According to the Supreme Court, the Examiner is required to provide an explicit analysis as to how the cited prior art teaches or suggests all the features of a claim. "To facilitate review, this [the Examiner's] analysis should be made explicit." (*KSR International Co. v. Teleflex, Inc.*, 550 U.S. ___ (2007). As demonstrated above, the Examiner has failed to specifically demonstrate how or where Sokolov teaches or suggests the features of the compilation of "native code output by said virtual machine." (*Id.*). Moreover, the Action has utterly failed to identify anywhere that Solokov teaches the "global optimizing compiler" recited in claims 25 and 35.

Therefore, under the standard of *KSR*, no *prima facie* case of obviousness has been made as to claims 25 and 35. For at least this reason, the rejection of claims 25 and 35 should be withdrawn.

Claims 27, 29, 37 and 39:

Claims 27, 29, 37 and 39 recite various limitations which related to the method for substituting a semantically enriched opcode for a corresponding code segment. Specifically, claims 27 and 37 recite modification of the virtual machine "by inserting a stub, said stub automatically loading a semantically enriched opcode when said virtual machine encounters an identified code segment." Claims 29 and 39 recite method/system in which a class loader substitutes semantically enriched opcodes for a corresponding code segment. The cited sections within Sokolov do not mention specific methods of making substitutions into computer code. Nowhere do Sokolov or Egashira teach or suggest "stubs" or "class loaders"

which make substitutions for code segments. For at least these additional reasons, the rejection of claims 29 and 39 should be reconsidered and withdrawn.

The recent Action states that "[t]hese claims are also rejected based on virtue of their dependencies on the rejected base claims 1 and 30, respectively." (Action, p. 6). Such a rejection is utterly unfounded. Nothing in the United States Code, the Code of Federal Regulations, or the Manual of Patent Examination and Procedure gives the Examiner authority to reject a dependent claim based on the rejection of its base independent claim. To the contrary, the MPEP explicitly states that "[i]f the base claim is rejected, the dependent claim should be objected to rather than rejected, if it is otherwise allowable." MPEP § 608.01(n)(II). Thus, dependent claims should be examined on their own merits, even if their corresponding base claims have been rejected. MPEP § 608.01(n)(III) recites:

> Examiners are reminded that a dependent claim is directed to a combination including everything recited in the base claim and what is recited in the dependent claim. *It is this combination that must be compared with the prior art, exactly as if it were presented as one independent claim.*

(Emphasis added).

Consequently, under MPEP § 608.01(n), the rejection of claims 27, 29, 37 and 39 is utterly inappropriate and beyond the authority granted to the Examiner. For at least this reason, the rejection should be reconsidered and withdrawn.

Conclusion:

In view of the foregoing arguments, all claims are believed to be in condition for allowance over the prior art of record. Therefore, this response is believed to be a complete response to the Office Action. However, Applicant reserves the right to set forth further arguments in future papers supporting the patentability of any of the claims, including the separate patentability of the dependent claims not explicitly addressed herein. In addition,

because the arguments made above may not be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed.

The absence of a reply to a specific rejection, issue or comment in the Office Action does not signify agreement with or concession of that rejection, issue or comment. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper, and the amendment of any claim does not necessarily signify concession of unpatentability of the claim prior to its amendment. Further, for any instances in which the Examiner took Official Notice in the Office Action, Applicants expressly do not acquiesce to the taking of Official Notice, and respectfully request that the Examiner provide an affidavit to support the Official Notice taken in the next Office Action, as required by 37 CFR 1.104(d)(2) and MPEP § 2144.03.

If the Examiner has any comments or suggestions which could place this application in better form, the Examiner is requested to telephone the undersigned attorney at the number listed below.

Respectfully submitted,

DATE: June 24, 2009

/Steven L. Nichols/
Steven L. Nichols
Registration No. 40,326

Steven L. Nichols, Esq.
Managing Partner, Utah Office
Rader Fishman & Grauer PLLC
River Park Corporate Center One
10653 S. River Front Parkway, Suite 150
South Jordan, Utah 84095

(801) 572-8066
(801) 572-7666 (fax)